(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 April 2002 (04.04.2002)

PCT

(10) International Publication Number
WO 02/27475 A2

(51) International Patent Classification[7]: G06F 9/302

(21) International Application Number: PCT/US01/30309

(22) International Filing Date:
26 September 2001 (26.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/675,066 28 September 2000 (28.09.2000) US

(71) Applicants: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US). **ANALOG DEVICES, INC.** [US/US]; One Technology Way, P.O. Box 9106, Norwood, MA 02062 (US).

(72) Inventors: **ROTH, Charles, P.**; 13305 Tichester Court, Austin, TX 78729 (US). **KALAGOTLA, Ravi**; 11500 Jolliville Road #1823, Austin, TX 78759 (US). **FRIDMAN, Jose**; 70 Centre Street, Apt.#5E, Brookline, MA 02446 (US).

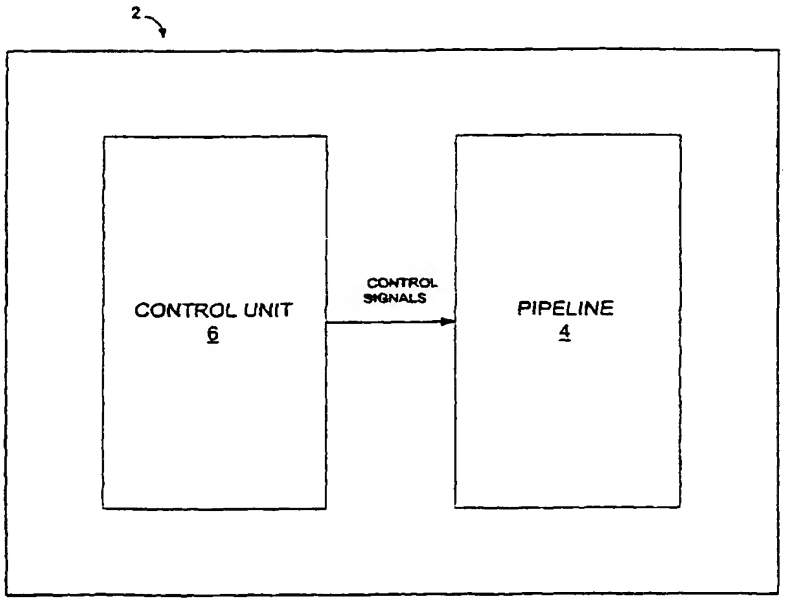
(74) Agent: **HARRIS, Scott, C.**; Fish & Richardson P.C., Suite 500, 4350 La Jolla Village Drive, San Diego, CA 92122 (US).

(81) Designated States *(national)*: CN, JP, KR, SG.

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: ARRAY SEARCHING OPERATIONS



(57) Abstract: In one embodiment, a programmable processor searches an array of N data elements in response to N/M machine instructions, where the processor has a pipeline configured to process M data elements in parallel. In response to the machine instructions, a control unit directs the pipeline to retrieve M data elements from the array of elements in a single fetch cycle, concurrently compare the data elements to M current extreme values, and update the current extreme values, as well as M references to the current extreme values, based on the comparisons.

WO 02/27475 A2

ARRAY SEARCHING OPERATIONS

BACKGROUND

This invention relates to array searching operations for a computer.

Many conventional programmable processors, such as digital signal processors (DSP), support a rich instruction set that includes numerous instructions for manipulating arrays of data. These operations are typically computationally intensive and can require significant computing time, depending upon the number of execution units, such as multiply-accumulate units (MACs), within the processor.

DESCRIPTION OF DRAWINGS

Figure 1 is a block diagram illustrating an example of a pipelined programmable processor according to the invention.

Figure 2 is a block diagram illustrating an example execution pipeline for the programmable processor.

Figure 3 is a flowchart for implementing an example array manipulation machine instruction according to the invention.

Figure 4 is a flowchart of an example routine for invoking the machine instruction.

Figure 5 shows a search instruction; and

Figure 6 shows N/M search instruction.

DESCRIPTION

Figure 1 is a block diagram illustrating a programmable processor 2 having an execution pipeline 4 and a control unit 6. Processor 2, as explained in detail below, reduces the computational time required by array manipulation operations. In particular, processor 2 may support a machine instruction, referred to herein as the SEARCH instruction, that reduces the computational time to search an array of numbers in a pipelined processing environment.

Pipeline 4 has a number of stages for processing instructions. Each stage processes concurrently with the other stages and passes results to the next stage in pipeline 4 at each clock cycle. The final results of each instruction emerge at the end of the pipeline in rapid succession.

Control unit 6 controls the flow of instructions and data through the various stages of pipeline 4. During the processing of an instruction, for example, control unit 6 directs the various components of the pipelined to fetch and decode the instruction, perform the corresponding operation and write the results back to memory or local registers.

Figure 2 illustrates an example pipeline 4 configured according to the invention. Pipeline 4, for example, has five stages: instruction fetch (IF), decode (DEC), address

calculation (AC), execute (EX) and write back (WB). Instructions are fetched from memory, or from an instruction cache, during the IF stage by fetch unit 21 and decoded within address registers 22 during the DEC stage. At the next clock cycle, the results pass to the AC stage, where data address generators 23 calculate any memory addresses that are necessary to perform the operation.

During the EX stage, execution units 25A through 25M perform the specified operation such as, for example, adding or multiplying numbers, in parallel. Execution units 25 may contain specialized hardware for performing the operations including, for example, one or more arithmetic logic units (ALU's), floating-point units (FPU) and barrel shifters. A variety of data can be applied to execution units 25 such as the addresses generated by data address generator 23, data retrieved from data memory 18 or data retrieved from data registers 24. During the final stage (WB), the results are written back to data memory or to data registers 24.

The SEARCH instruction supported by processor 2, may allow software applications to search an array of N data elements by issuing N/M search instructions, where M is the number of data elements that can be processed in parallel by execution units 25 of pipeline 4. Note, however, that a single execution unit may be capable of executing two or more operations in parallel. For example, an execution unit

may include a 32-bit ALU capable of concurrently comparing two 16-bit numbers.

Generally, the sequence of SEARCH instructions allow the processor to process M sets of elements in parallel to identify an ``extreme value'', such as a maximum or a minimum, for each set. During the execution of the search instructions, processor 2 stores references to the location of the extreme value of each of the M sets of elements. Upon completion of the N/M instructions, as described in detail below, the software application analyzes the references to the extreme values for each set to quickly identify an extreme value for the array. For example, the instruction allows the software applications to quickly identify either the first or last occurrence of a maximum or minimum value. Furthermore, as explained in detail below, processor 2 implements the operation in a fashion suitable for vectorizing in a pipelined processor across the M execution units 25.

As described above, a software application searches an array of data by issuing N/M SEARCH machine instructions to processor 2. Figure 3 is a flowchart illustrating an example mode of operation 20 for processor 2 when it receives a single SEARCH machine instruction. Process 20 is described with reference to identifying the last occurrence of a minimum value within the array of elements; however, process 20 can be easily modified to perform other functions

such as identifying the first occurrence of a minimum value, the first occurrence of a maximum value or a last occurrence of a maximum value.

For exemplary purposes, process 20 is described in assuming M equals 2, i.e., processor 2 concurrently processes two sets of elements, each set having N/2 elements. However, the process is not limited as such and is readily extensible to concurrently process more than two sets of elements. In general, process 20 facilitates vectorization of the search process by fetching pairs of elements as a single data quantity and processing the element pairs through pipeline 4 in parallel, thereby reducing the total number of clock cycles necessary to identify the minimum value within the array. Although applicable to other architectures, process 20 is well suited for a pipelined processor 2 having multiple execution units in the EX stage. For each set of elements, process 20 maintains two pointer registers, P_{Even} and P_{Odd}, that store locations for the current extreme value within the corresponding set. In addition, process 20 maintains two accumulators, A0 and A1,that hold the current extreme values for the sets. The pointer registers and the accumulators, however, may readily be implemented as general-purpose data registers without departing from process 30.

Referring to Figure 3, in response to each SEARCH instruction, processor 2 fetches a pair of elements in one

clock cycle as a single data quantity (21). For example, processor 2 may fetch two adjacent 16-bit values as one 32-bit quantity. Next, processor 2 compares the even element of the pair to a current minimum value for the even elements (22) and the odd element of the pair to a current minimum value for the odd elements (24).

When a new minimum value for the even elements is detected, processor 2 updates accumulator A0 to hold the new minimum value and updates a pointer register P_{Even} to hold a pointer to corresponding data quantity within the array (23). Similarly, when a new minimum value for the odd elements has been detected, processor 2 updates accumulator A1 and a pointer register P_{Odd} (25). In this example, each pointer register P_{Even} and P_{Odd} points to the data quantity and not the individual elements, although the process is not limited as such. Processor 2 repeats the process until all of the elements within the array have been processed (26). Because processor 2 is pipelined, element pairs may be fetched until the array is processed.

The following illustrates exemplary syntax for invoking the machine instruction:

$$(P_{Odd}, P_{Even}) = \text{SEARCH } R_{Data} \text{ LE, } R_{Data} = [P_{fetch\_addr}++]$$

Data register R_{Data} is used as a scratch register to store each newly fetched data element pair, with the least significant word of R_{Data} holding the odd element and the

most significant word of R_{Data} holding the even element. Two accumulators, A0 and A1, are implicitly used to store the actual values of the results. An additional register, P_{fetch_addr}, is incremented when the SEARCH instruction is issued and is used as a pointer to iterate over the N/2 data quantities within the array. The defined condition, such as ``less than or equal'' (LE) in the above example, controls which comparison is executed and when the pointer registers $P_{Even}$ and $P_{Odd}$, as well as the accumulators A0 and A1, are updated. The ``LE'', for example, directs processor 2 to identify the last occurrence of the minimum value.

In a typical application, a programmer develops a software application or subroutine that issues the N/M search instructions, probably from within a loop construct. The programmer may write the software application in assembly language or in a high-level software language. A compiler is typically invoked to processes the high-level software application and generate the appropriate machine instructions for processor 2, including the SEARCH machine instructions for searching the array of data.

Figure 4 is a flowchart of an example software routine 30 for invoking the example machine instructions illustrated above. First, the software routine 30 initializes the registers including initializing A0 and A1 and pointing P_{Eve} and P_{Odd} to the first data quantity within the array (31). In one embodiment, software routine 30 initializes a loop

count register with the number of SEARCH instructions to issue (N/M). Next, routine 30 issues the SEARCH machine instruction N/M times. This can be accomplished a number of ways, such as by invoking a hardware loop construct supported by processor 2. Often, however, a compiler may unroll a software loop into a sequence of identical SEARCH instructions (32).

After issuing N/M search instructions, A0 and A1 hold the last occurrence of the minimum even value and the last occurrence of the minimum odd value, respectively. Furthermore, P_{Even} and P_{Odd} store the locations of the two data quantities that hold the last occurrence of the minimum even value and the last occurrence of the minimum odd value.

Next, in order to identify the last occurrence of the minimum value for the entire array, routine 30 first increments P_{Odd} by a single element, such that P_{Odd} points directly at the minimum odd element (33). Routine 30 compares the accumulators A0 and A1 to determine whether the accumulators contain the same value, i.e., whether the minimum of the odd elements equals the minimum of the even elements (34). If so, the routine 30 compares the pointers to determine whether P_{Odd} is less than P_{Even} and, therefore, P_{Odd} and P_{Even} whether the minimum even value occurred earlier in the array (35). Based on the comparison, the routine determines whether to copy P_{Odd} into P_{Even} (37).

When the accumulators A0 and A1 are not the same, the routine compares A0 to A1 in order to determine which holds the minimum value (36). If A1 is less than A0 then routine 30 sets P_{Even} equal to P_{Odd}, thereby copying the pointer to the minimum value from P_{Odd} into P_{Even} (37).

At this point, P_{Even} points to the last occurrence of the minimum value for the entire array. Next, routine 30 adjusts P_{Even} to compensate for errors introduced to the pipelined architecture of processor 2 (38). For example, the comparisons described above are typically performed in the EX stage of pipeline 4 while incrementing the pointer register P_{fetch_addr} typically occurs during the AC stage, thereby causing the P_{Odd} and P_{Even} to be incorrect by a known quantity. After adjusting P_{Even}, routine 30 returns P_{Even} as a pointer to the last occurrence of the minimum value within the array(39).

Figure 5 illustrates the operation for a single SEARCH instruction as generalized to the case where processor 2 is capable of processing M elements of the array in parallel, such as when processor 2 includes M execution units. The SEARCH instruction causes processor 2 to fetch M elements in a single fetch cycle (51). Furthermore, in this example, processor 2 maintains M pointer registers to store addresses (locations) of a corresponding extreme value for each of the M sets of elements. After fetching the M elements, processor 2 concurrently compares the M elements to a

current extreme value for the respective element set, as stored in M accumulators (52). Based on the comparisons, processor 2 updates the M accumulators and the M pointer registers (53).

Figure 6 illustrates the general case where a software application issues N/M SEARCH instructions and, upon completion of the instructions, determines the extreme value for the entire array. First, the software application initializes a loop counter, the M accumulators used to store the current extreme values for the M element sets and the M pointers used to store the locations of the extreme values (61). Next, the software application issues N/M SEARCH instructions (62). After completion of the instructions, the software application may adjust the M pointer registers to correctly reference its respective extreme value, instead of the data quantity holding the extreme value (63). After adjusting the pointer registers, the software application compares the M extreme values for the M element sets to identify an extreme value for the entire array, i.e., a maximum value or a minimum value (64). Then, the software application may use the pointer registers to determine whether more than one of the element sets have an extreme value equal to the array extreme value and, if so, determine which extreme value occurred first, or last, depending upon the desired search function (65).

Various embodiments of the invention have been described. For example, a single machine instruction has been described that searches an array of data in a manner that facilitates vectorization of the search process within a pipelined processor. The processor may be implemented in a variety of systems including general purpose computing systems, digital processing systems, laptop computers, personal digital assistants (PDA's) and cellular phones. For example, cellular phones often maintain an array of values representing signal strength for services available 360° around the phone. In this context, the process discussed above can be readily used upon initialization of the cellular phone to scan the available services and quickly select the best service. In such a system, the processor may be coupled to a memory device, such as a FLASH memory device or a static random access memory (SRAM), that stores an operating system and other software applications. These and other embodiments are within the scope of the following claims.

What is claimed is:

1. A method comprising:

receiving a machine instruction directing a processor to search a plurality of data elements; and

executing the machine instruction by:

retrieving M data elements in a single fetch cycle;

concurrently comparing the M data elements to a corresponding current extreme value; and

updating a set of references based on the comparisons.

2. The method of claim 1, wherein retrieving the M data elements comprises retrieving the M data elements as a single data quantity containing the M data elements.

3. The method of claim 2, wherein the set of references comprise pointer registers to store addresses for data quantities.

4. The method of claim 1, wherein M = 1.

5. The method of claim 1, wherein M = 2.

6. The method of claim 1, wherein executing the machine instruction further includes:

storing the current extreme values in M accumulators; and

copying the M data elements to the accumulators based on the comparisons.

7. The method of claim 5, wherein concurrently comparing the data elements comprises processing a first data element with a first execution unit of a pipelined processor and processing a second data element with a second execution unit of the pipelined processor.

8. The method of claim 5, wherein concurrently comparing the data elements comprises concurrently processing a first data element and a second data element within a single execution unit of a pipelined processor.

9. The method of claim 1, wherein concurrently comparing each of the data elements to a current extreme value includes determining whether each of the data elements is less than the corresponding current extreme value.

10. The method of claim 1, wherein concurrently comparing each of the data elements to a current extreme value includes determining whether each of the data elements is greater than the corresponding current extreme value.

11. A method for searching an array of N data elements for a value comprising:

issuing N/M machine instructions to a processor, wherein the processor is adapted to process M data elements in parallel; and

analyzing results of the machine instructions to identify a value for the array.

12. The method of claim 11, further comprising:

executing each machine instruction by:

retrieving M data elements in a single fetch cycle,

concurrently comparing each of the M data elements to a corresponding current extreme value, and

updating the references based on the comparisons.

13. A method comprising:

retrieving the pair of data elements from an array of elements in a single fetch operation, wherein the pair of data elements includes an even data element and an odd data element;

substantially comparing the even element of the pair and the odd element of the pair; and

substantially fetching and comparing the remaining pairs of data elements of the array until all of the data elements of the array have been processed.

14. The method of claim 13, wherein substantially comparing the pair of data elements includes setting an even minimum value as function of the even element of the element pair

and setting an odd minimum value as function of the odd element of the element pair.

15. The method of claim 13, wherein substantially comparing the pair of data elements includes maintaining a first accumulator to store a minimum value for the even elements and a second accumulator to store a minimum value for the odd elements.

16. The method of claim 13, further including maintaining a first pointer register to store an address for the minimum value of the even data elements and maintaining a second pointer register to store an address for the minimum value of the odd data elements.

17. The method of claim 16, further including adjusting at least one of the pointer registers after processing all of the pairs of data elements to account for a number of stages in the pipeline.

18. The method of claim 13, wherein the method is invoked by issuing N/M machine instructions to a programmable processor, wherein N equals the number of elements in the array and M equals the number of data elements that the processor can concurrently compare.

19. An apparatus comprising:

a pipeline adapted to process M data elements in parallel; and

a control unit adapted to direct the execution pipeline to search an array of N data elements in response to N/M machine instructions.

20. The apparatus of claim 19, wherein in response to the machine instructions, the control unit directs the pipeline to retrieve M data elements from the array of elements in a single fetch operation and concurrently compare the data elements to a corresponding current extreme value.

21. The apparatus of claim 19, wherein the pipeline includes M registers adapted to store references to the extreme values.

22. The apparatus of claim 21, wherein the registers are pointer registers.

23. The apparatus of claim 21, wherein the registers are general-purpose data registers.

24. The apparatus of claim 18, wherein the pipeline includes M accumulators to store M current extreme values.

25. The apparatus of claim 18, wherein the pipeline includes M general-purpose registers to store M current extreme values.

26. An article comprising a medium having computer-executable instructions stored thereon for compiling a software program, wherein the computer-executable instructions are adapted to generate N/M machine instructions to search an array of N data elements, each machine instruction causing a programmable processor to:

retrieve M data elements from an array of N elements in a single fetch operation; and

sustantially compare each of the M data elements to a corresponding current extreme value.

27. The article of claim 26, wherein each machine instruction causes the processor to update a set of references based on the comparisons.

28. The article of claim 26, wherein each machine instruction causes the processor to concurrently process a first data element and a second data element within a single execution unit of a pipelined processor.

29. A system comprising:

a memory device; and

a processor coupled to the memory device, wherein the processor includes a pipeline configured to process M data elements in parallel and a control unit configured to

direct the pipeline to search an array of N data elements in response to N/M machine instructions.

30. The system of claim 29, wherein in response to each machine instructions, the control unit directs the pipeline to retrieve M data elements from the array of elements in a single fetch operation and concurrently compare the data elements to a corresponding current extreme value.

31. The system of claim 29, wherein the pipeline includes M registers configured to store references to the extreme values.

32. The system of claim 31, wherein the registers are pointer registers.

33. The system of claim 31, wherein the registers are general-purpose data registers.

34. The system of claim 29, wherein the memory device comprises static random access memory.

35. The system of claim 29, wherein the memory device comprises FLASH memory.

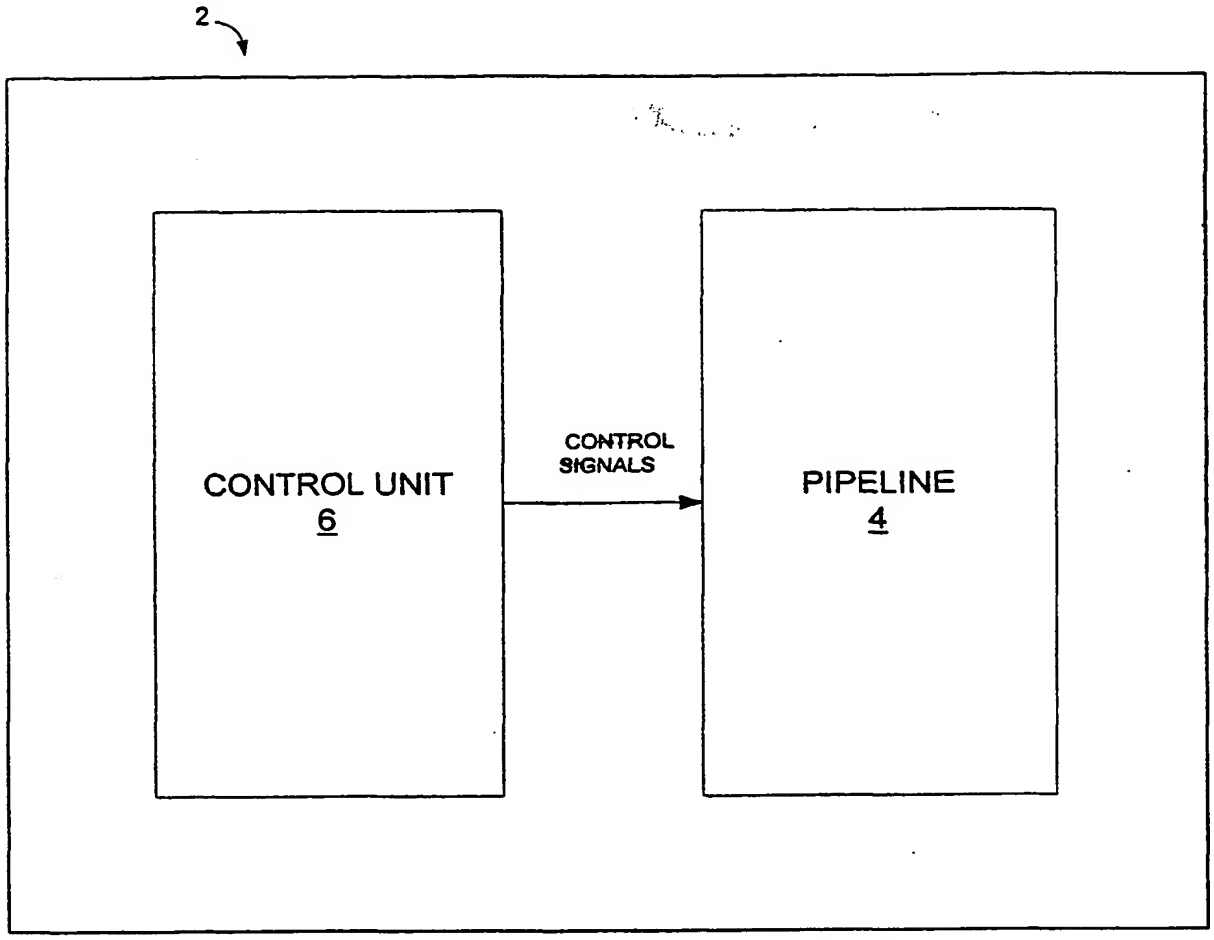


FIG. 1

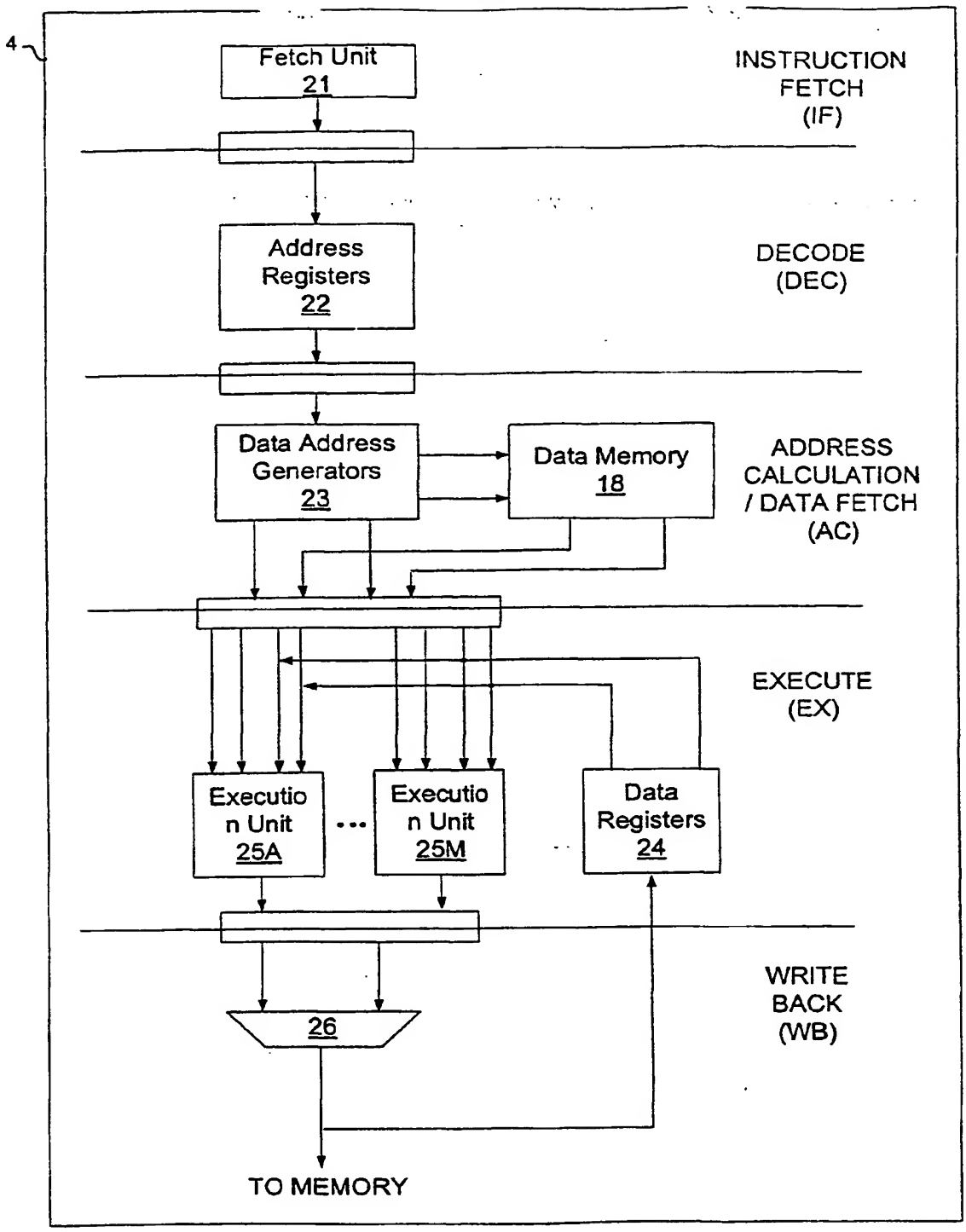


FIG. 2

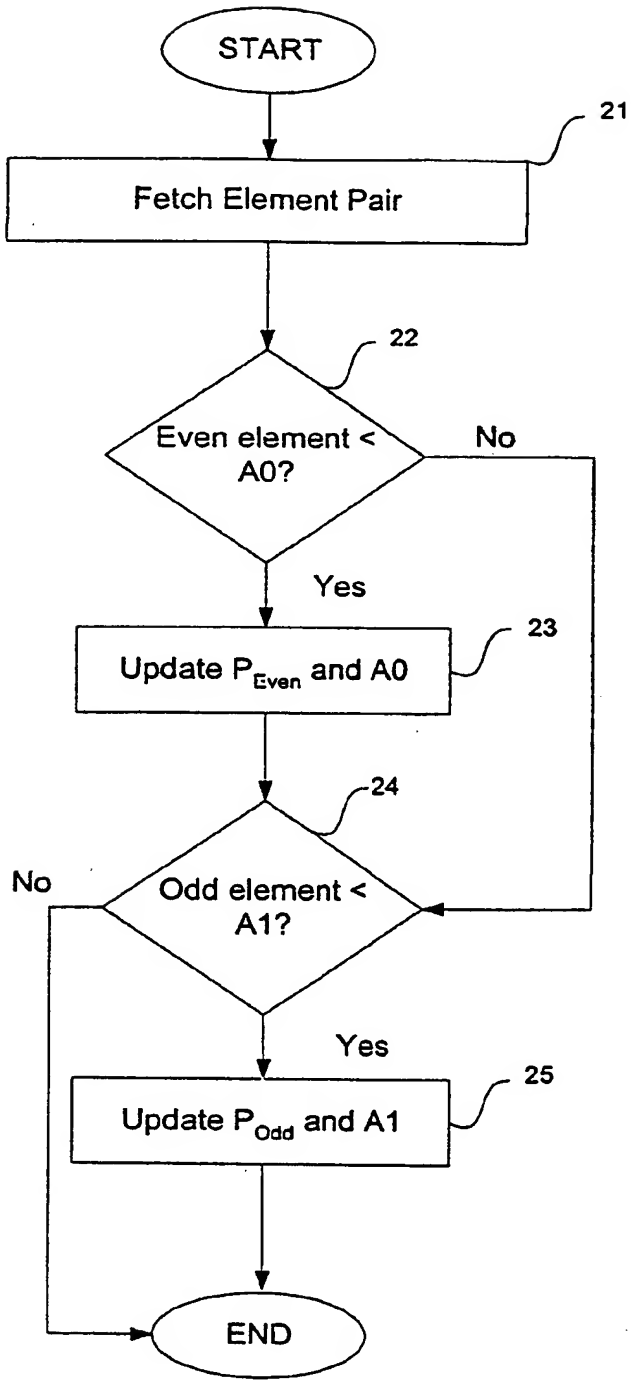

20
START
21
Fetch Element Pair
22
Even element < A0?
No
Yes
23
Update P_{Even} and A0
24
No
Odd element < A1?
Yes
25
Update P_{Odd} and A1
END

FIG. 3

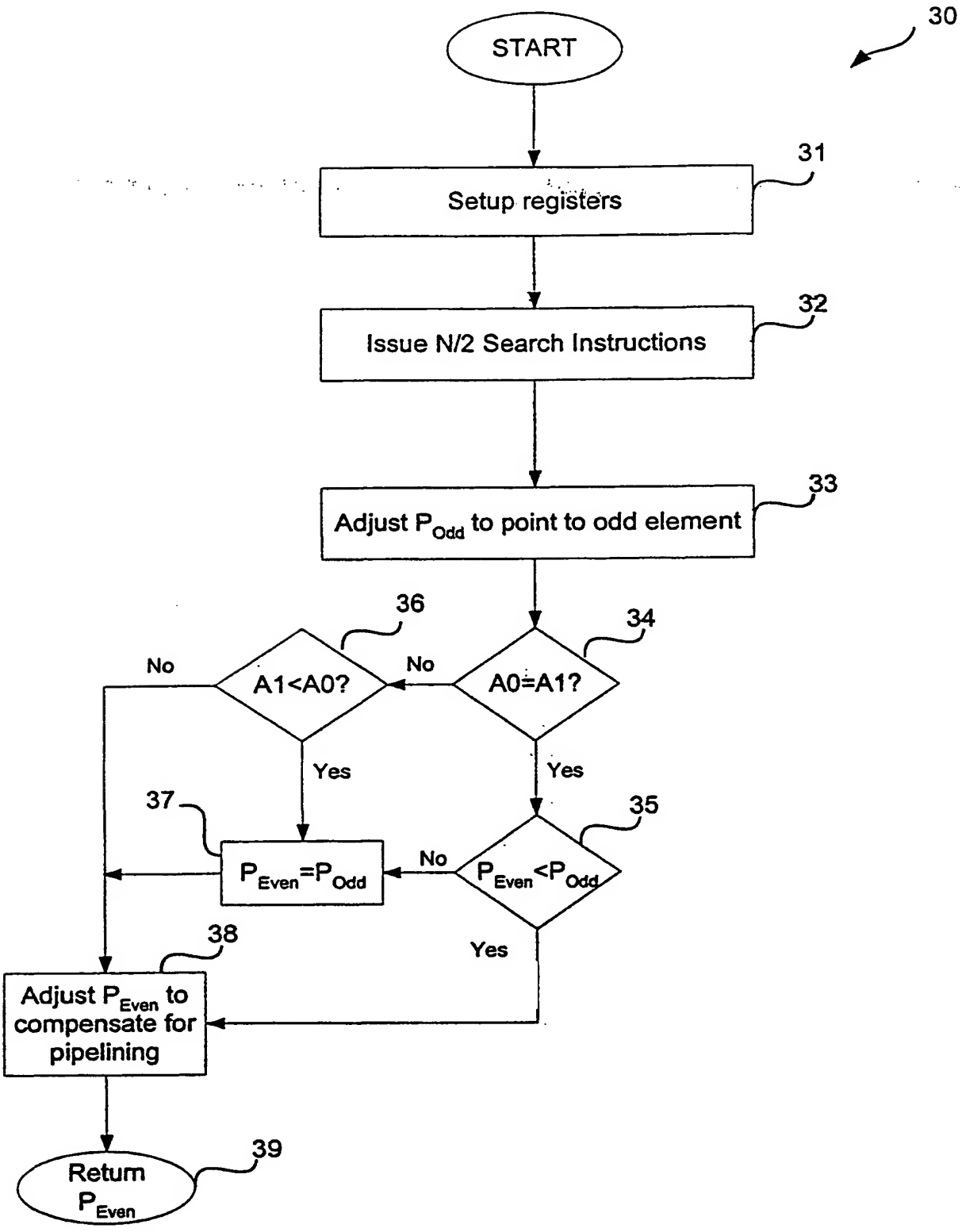


FIG. 4

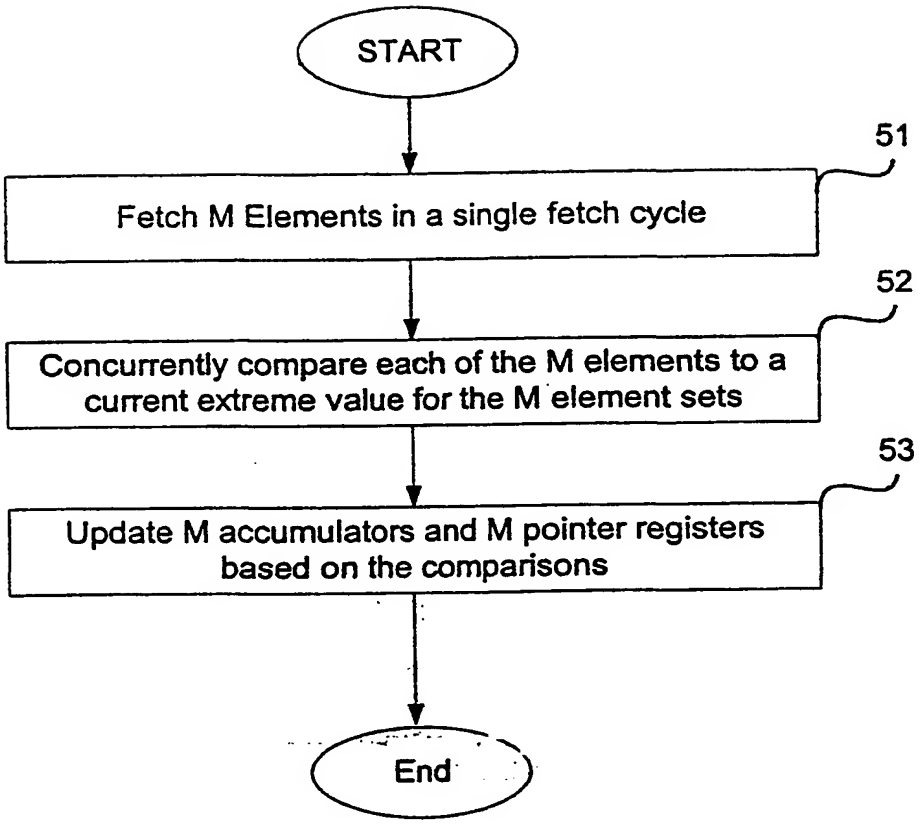

START
51
Fetch M Elements in a single fetch cycle
52
Concurrently compare each of the M elements to a current extreme value for the M element sets
53
Update M accumulators and M pointer registers based on the comparisons
End

FIG 5

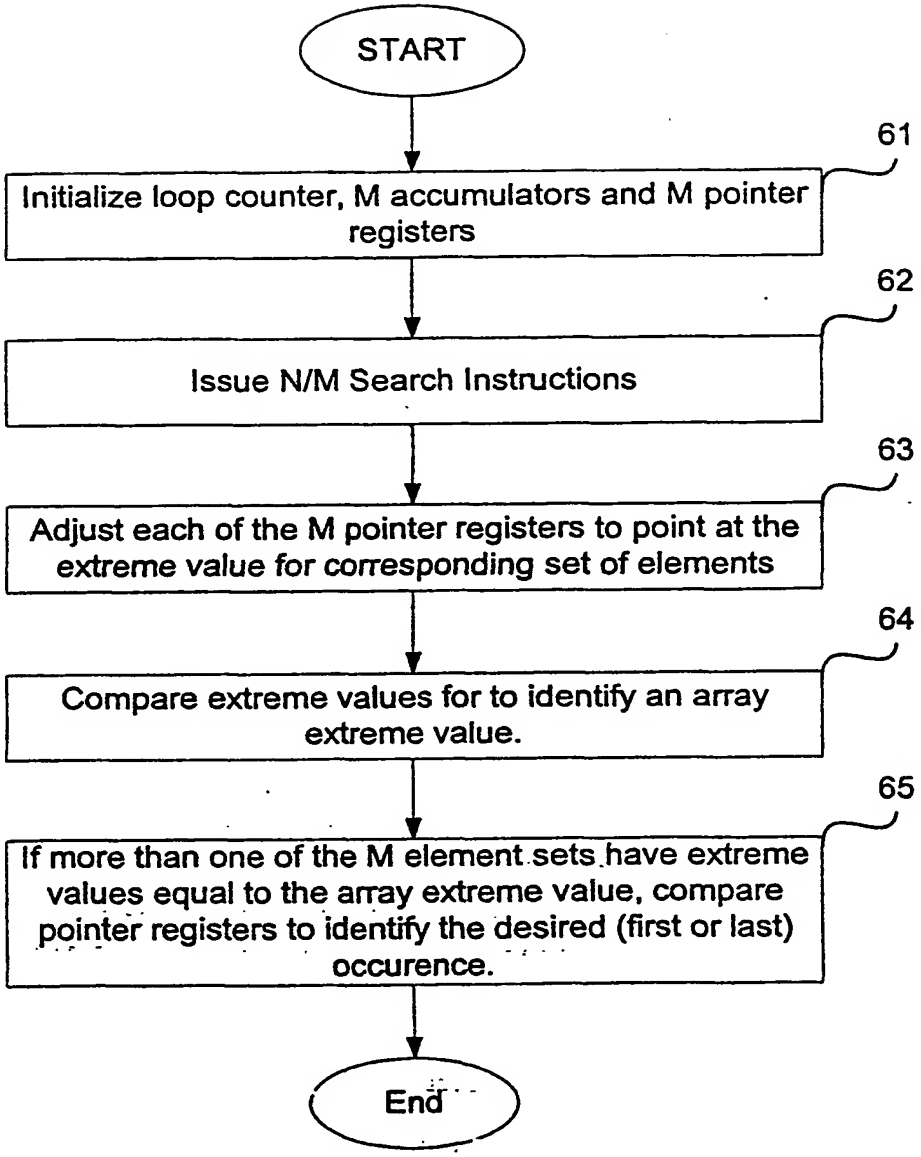


FIG. 6

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 April 2002 (04.04.2002)

PCT

(10) International Publication Number
WO 02/27475 A3

(51) International Patent Classification[7]: G06F 9/302, 9/30, 7/22

(21) International Application Number: PCT/US01/30309

(22) International Filing Date:
26 September 2001 (26.09.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/675.066 28 September 2000 (28.09.2000) US

(71) Applicants: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US). **ANALOG DEVICES, INC.** [US/US]; One Technology Way, P.O. Box 9106, Norwood, MA 02062 (US).

(72) Inventors: **ROTH, Charles, P.**; 13305 Tichester Court, Austin, TX 78729 (US). **KALAGOTLA, Ravi**; 11500 Jolliville Road #1823, Austin, TX 78759 (US). **FRIDMAN, Jose**; 70 Centre Street, Apt.#5E, Brookline, MA 02446 (US).

(74) Agent: **HARRIS, Scott, C.**; Fish & Richardson P.C., Suite 500, 4350 La Jolla Village Drive, San Diego, CA 92122 (US).

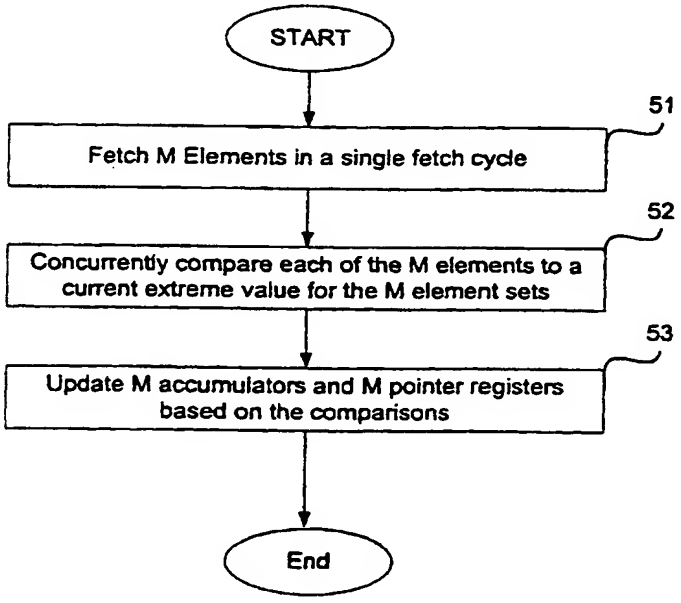
(81) Designated States *(national)*: CN, JP, KR, SG.

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(88) Date of publication of the international search report:
13 June 2002

[Continued on next page]

(54) Title: ARRAY PROCESSING OPERATIONS



(57) Abstract: In one embodiment, a programmable processor searches an array of N data elements in response to N/M machine instructions, where the processor has a pipeline configured to process M data elements in parallel. In response to the machine instructions, a control unit directs the pipeline to retrieve M data elements from the array of elements in a single fetch cycle, concurrently compare the data elements to M current extreme values, and update the current extreme values, as well as M references to the current extreme values, based on the comparisons.

WO 02/27475 A3

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/30309

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/302 G06F9/30 G06F7/22

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 4 774 688 A (KOBAYASHI MAKOTO ET AL) 27 September 1988 (1988-09-27) the whole document	1,2,4,6, 9,10 5,7,8, 11-15, 18-21, 23-31, 33-35

Y	WO 99 23548 A (ADVANCED MICRO DEVICES INC) 14 May 1999 (1999-05-14) page 51, paragraph 8 -page 52, line 10 page 53, line 20 -page 55, line 14 page 55, line 31 -page 56, line 17	5,7,8, 11-15, 18-21, 23-31, 33-35
	--- -/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 March 2002	09/04/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Daskalakis, T

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 01/30309

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5 187 675 A (DENT PAUL W ET AL) 16 February 1993 (1993-02-16) column 5, line 8 - line 49	3,16,22, 32
A	"ALU Implementing Native Minimum/Maximum Function for Signal Processing Applications" IBM TECHNICAL DISCLOSURE BULLETIN, IBM CORP. NEW YORK, US, vol. 5, no. 29, 1 October 1986 (1986-10-01), pages 1975-1978, XP002079689 ISSN: 0018-8689	

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 01/30309

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4774688 A	27-09-1988	JP 61122747 A	10-06-1986
		EP 0181516 A2	21-05-1986
WO 9923548 A	14-05-1999	US 6256653 B1	03-07-2001
		US 6131104 A	10-10-2000
		US 6085212 A	04-07-2000
		US 6094668 A	25-07-2000
		US 6088715 A	11-07-2000
		US 6085208 A	04-07-2000
		US 6298367 B1	02-10-2001
		US 6223192 B1	24-04-2001
		DE 69801678 D1	18-10-2001
		EP 1061436 A2	20-12-2000
		EP 1031073 A2	30-08-2000
		EP 1025485 A2	09-08-2000
		US 6026483 A	15-02-2000
		US 5918062 A	29-06-1999
		US 6085213 A	04-07-2000
		US 6038583 A	14-03-2000
		US 6115733 A	05-09-2000
		WO 9921078 A2	29-04-1999
		WO 9923548 A2	14-05-1999
		US 6029244 A	22-02-2000
		US 2001051969 A1	13-12-2001
US 5187675 A	16-02-1993	AU 651737 B2	28-07-1994
		AU 2656392 A	27-04-1993
		BR 9205409 A	15-03-1994
		DE 4293456 T0	07-10-1993
		ES 2100114 A1	01-06-1997
		FR 2681450 A1	19-03-1993
		GB 2265033 A ,B	15-09-1993
		HK 27196 A	23-02-1996
		IT 1255822 B	17-11-1995
		JP 6505587 T	23-06-1994
		MX 9205325 A1	01-08-1993
		NL 9220005 T	01-09-1993
		SE 515269 C2	09-07-2001
		SE 9301546 A	28-06-1993
		WO 9306547 A1	01-04-1993